= = = Be sure to read this note = = =

M32R Family C/C++ compiler package (CC32R)

V.5.01 Release 00

Release Notes 4th Edition

Renesas Solutions Corp.

September 10, 2008

Welcome to M32R Family C / C++ compiler package (abbreviated as CC32R) V.5.01 Release 00. This document contains supplementary descriptions to the electronic User's Manual. Please read this release note while you refer to a corresponding item in electronic User's Manual.

Renesas Technology Corp. and Renesas Solutions Corp. reserve the right to change the contents of this release note without notice for improvements on characteristics, etc.

Active X, Microsoft, MS-DOS, Visual Basic, Visual C++, Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

HP-UX is a registered trademark of Hewlett-Packard Company.

Sun, Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. or other countries, and are used under license.

UNIX is a registered trademark of The Open Group in the United States and other countries.

IBM and AT are registered trademarks of International Business Machines Corporation.

HP9000 is a product name of Hewlett-Packard Company.

SPARC and SPARCstation are registered trademarks of SPARC International, Inc.

Intel and Pentium are registered trademarks of Intel Corporation.

i386, i486, and MMX are trademarks of Intel Corporation.

Adobe and Acrobat are registered trademarks of Adobe Systems Incorporated.

Netscape and Netscape Navigator are registered trademarks of Netscape Communications Corporation in the U.S. and other countries.

All other brand and product names are trademarks, registered trademarks or service marks of their respective holders.

==========

COPYRIGHT(C) 2008 **Renesas Technology Corp.** and **Renesas Solutions Corp.**

1. Changes from the previous version

The changes from V.5.00 Release 00 are as follows.

Some part of the C99 specification were appended (C/C++ compiler):

The next three items of C99 language specification(ISO / IEC 9899:1999) were appended.

- Designation
  It is a new function in an initializer-list. That indicates a initializer is for which element of a struct or an array.
- Pragma Operator
  It is like a function style macro and has the same means of a #pragma directive.
- Variable Argument Macro
  A function-like macro definition that has uncertain number of arguments can be declared.

Place of the library generator was changed:

The library generator storing place was changed to the “bin32R” directory instead of the “lib32R” directory.

The eight problems as follows have been fixed:

- On expanding an inline function, some of branch and jump machine-codes has the incorrect targets, or an internal error occurs (C / C++ compiler):
  Published to RENESAS TOOL NEWS on April 16, 2007: RSO-M3T-CC32R-070416D
- On compiling with -cs option an inline-asm function, the compiler does not output the content of this asm-function to the object files. (C / C++ compiler):
  Published to RENESAS TOOL NEWS on March 16, 2007: RSO-M3T-CC32R-070316D
- On comparing a pointer with an integer of 0 using the != operator, the optimizer mistakes the operation as always 1 (C / C++ compiler):
  Published to RENESAS TOOL NEWS on April 16, 2006: RSO-M3T-CC32R-060416D
- On re-linking a relocatable load module file, an link error occurs (C / C++ compiler):
  Published to RENESAS TOOL NEWS on March 16, 2006: RSO-M3T-CC32R-060316D
- On constant expressions of integers containing a shift operator whose right operand is an unsigned integer type, an compile error occurs (C / C++ compiler):
  Published to RENESAS TOOL NEWS on March 1, 2006: RSO-M3T-CC32R-060301D
- On declaring the template variable as an array, an error may occur at the linking (C / C++ Compiler)
- On dividing the minimum value of the long type by minus one, CC32R will show an error (C / C++ Compiler)
- On masking the members in the same union with a constant of the range of 0x80000000-0xffffffff, internal error will occur (C / C++ Compiler)

Erratas of User's Manuals

User's manuals have some mistypes. So, please read these manuals while correcting to the contents as following table.

<< C / C++ Compiler >>

Page	Place	Before Correction	After Correction
CC32R MANUAL-vi (6 / 465)	Chapter 7 Embedded Applications Programming	7.4 About start-up file start.ms in HEW	7.4 About start-up file start.ms in High-performance Embedded Workshop

(To be continued to the next page)

(Continued from the previous page)

Page	Place	Before Correction	After Correction
CC32R MANUAL-13 (27/465)	2.2 Compatibility with an old version	(N/A)	Map output during link error supported (-MAP option): Even when an error occurs during a link operation where a map generation (cc32R: -MAP option, lnk32R: -M option) operation is specified, a temporary link map file will now be generated. Although this is essentially temporary information because the addresses in it differ from those generated during normal link, it will prove effective when analyzing the cause of a link error that occurred for reasons of section allocation, etc.
CC32R MANUAL-15 (29/465)	3.1.3 Command Line Syntax and Rules Figure 3.1 cc32R Command Line Syntax	-warn_suppressed_nested_comment	-warn_suppress_nested_comment
CC32R MANUAL-140 (154/465)	7.4 (The title of this section)	7.4 About start-up file start.ms in HEW	7.4 About start-up file start.ms in High-performance Embedded Workshop
CC32R MANUAL-140 (154/465)	7.4 About start-up file start.ms in HEW Main text	The HEW generates a file "start.ms" when creating new project. This file was modified from one that was using with the TM and the User's Manual.	The High-performance Embedded Workshop generates a file "start.ms" when creating new project. This file was modified from one that was using with the TM and the User's Manual.
CC32R MANUAL-140 (154/465)	7.4 About start-up file start.ms in HEW Main text	Fundamentally, the contents of these start.ms are nearly equal. However, the start.ms HEW generated can be controled by the assembler as32R with setting following parameter into -D option. If you will modify this start.ms, be careful in this point.	Fundamentally, the contents of these start.ms are nearly equal. However, the start.ms High-performance Embedded Workshop generated can be controled by the assembler as32R with setting following parameter into -D option. If you will modify this start.ms, be careful in this point.
CC32R MANUAL-140 (154/465)	7.4 About start-up file start.ms in HEW Table 7.3. (The title of this table)	Table 7.3. Meaning of control symbols of start.ms the HEW generated	Table 7.3. Meaning of control symbols of start.ms the High-performance Embedded Workshop generated
CC32R MANUAL-140 (154/465)	7.4 About start-up file start.ms in HEW Table 7.3. Meaning of control symbols of start.ms the HEW generated	Item name of HEW project Name creating dialog	Item name of High-performance Embedded Workshop project Name creating dialog
CC32R MANUAL-328 (342/465)	10.2.8 Registers ANSI-C 6.5.1 Storage-class specifiers <The extent to which objects can actually be placed in registers by use of the register storage-class specifier.>	The register storage-class specifier is ignored.	The register storage-class specifier will be accepted, but the all register specified objects may not be allocated for registers. A compiler judges which object should be allocated a register. There is no limit to the number of register declarations.

2. Precautions to be observed when installing

Please pay attention to the following point, in starting installation.

About the TM

The TM, older integrated environment, cannot be used in CC32R V.5.00 Release 00 or later.

The integrated environment for CC32R had been changed to the High-performance Embedded Workshop (it is already included in this product). Therefore, please use it.

[About the TM]
http://www.renesas.com/fmwk.jsp?cnt=ide_tm_tools_product_landing.jsp&fp=/products/tools/ide/ide_tm/

[About the High-performance Embedded Workshop]
http://www.renesas.com/fmwk.jsp?cnt=ide_hew_tools_product_landing.jsp&fp=/products/tools/ide/ide_hew/

For reference, about how to move to the High-performance Embedded Workshop, please refer the URL as follows.

[A guide to Porting Projects Created with TM to High-performance Embedded Workship V.4. (PDF,352 KB)]
http://www.renesas.com/media/support/faqimage/products/mpumcu/tool/hew_106629_en_GL_pdf.pdf

About the license ID

You need to input a license ID in the intermediate step of installation. Before you start installing CC32R, check your license ID.

Yet, a fee is necessary to the changing from previous version to V.5.00 Release 00. Therefore, this version of CC32R cannot be installed by license ID of V.4.30 Release 00 or earlier.

Necessary memory and HDD capacity

In order that CC32R operates comfortably, it requires at least 64 Mbytes of memory and a hard disk drive having 500M bytes or more bytes of space.

Operating environment

The CC32R does not run under Windows 3.1 and Windows NT3.5x or earlier.

Precautions about the file name and directory name

The file name and directory name of the source program must follow the precautions described below:

- Directory and file names that contain multi-byte character (ex. Japanese-Kanji) cannot be used.
- Only one priod (.) can be used in a file name.
- Network path names cannot be used. Assign the path to a drive name.
- Shortcuts cannot be used.
- Directory and file names that contain a space character cannot be used.
- The "..." symbol cannot be used as a means of specifying two or more directories.
- A file name in length of 128 characters or more including path specification cannot be used.

When should the SQMlint be installed (MISRA C Rule Checker).

Please be sure to install the SQMlint after the CC32R.

If the CC32R is installed after the SQMlint, that cannot use the function of the SQMlint.

For details, refer to the manual or release note that are attached to the SQMlint.

About the Install Manager (V.5.01 or later)

The CC32R that is the V.5.01 Release 00 or later is installed by through the Install Manager. Therefore, please confirm the paragraph 3.1 "How to install CC32R" in this document,

because the installation procedure differs from the V.5.00 Release 00 and older.

The Install Manager can install plural integrated environment High-performance Embedded Workshop on one PC. You can construct plural tool-environments (means combinations of compilers and debuggers) by this function of the Install Manager.

For more details, look the help topics that will be displayed when first Install Manager execution.

About technical supporting

For technical supporting, input user information in an installing. Then, the support information will be displayed in the support information tool on end of installations.

Please tell us question item with this support information when you take a technical support.

For information on our policy concerning the protection of personal information, please refer to the Renesas Technology Homepage.

URL: http://www.renesas.com/fmwk.jsp?cnt=privacy.htm&fp=/privacy/&site=i

The information we receive via the User Registration Form aids us greatly in our customer support activities. We provide Renesas Technology and related companies, distributors, etc., with essential user information (electronically or on paper) that will further help them provide customer support.

If you do not wish to have your user information provided to other related companies, please contact us to let us know. Note, however, this will limit our ability to provide complete product support.

Settings of environment variables

The environment variables listed below must always be set. If one of these variables remains unset or an invalid path is specified, an error may occur when running the software. (These variables do not need to be set when running the CC32R from TM.)

```
M32RLIB
M32RBIN
M32RINC
M32RTMP
```

Precautions about virus check programs

If this software is started while a virus check program is resident in memory, it may not operate properly. In such a case, remove the virus check program from memory before you start the software.

Restrictions and Precautions

The Restrictions and Precautions sometimes are being published to this document. Before using CC32R please read attentively these contents.

These contents include several precautions that were found after the manual was made.

How to get the latest information of CC32R

Renesas has an internet home page in the URL shown below. The latest information on Renesas Development Environment are published here.

```
http://www.renesas.com/en/tools
```

3. Installing CC32R

3.1 How to install CC32R

1. Execute the installer by the method of items as follows (A or B).

A. In the case of installing from the included CD:
Set the CD to CD-ROM drive of your PC. The installer (install manager) will run automatically. If it will not run automatically, please run the `hewinstman.exe` on the CD root directory.

B. In the case of utilizing online version up]:
Download installer files and install by the method that is written to the tool homepage.

2. Control the install manager by the method of items as follows (A or B).

A. In the case of the first time of High-performance Embedded Workshop installing:
✶ If the High-performance Embedded Workshop is already installed → Jump to "B. The case of High-performance Embedded Workshop already installed"

1) Push "Installation" button.

[About entering user information]
The data you input in the intermediate of installation is used to create a file for technical supporting sheet.

❋ Please confirm "Entering user registration" of chapter 2, for information on our policy concerning the protection of personal information.

2) Check "Install a High-performance embedded workshop for the first time." and push "Next" button.

3) The install directory of High-performance Embedded Workshop will be displayed on the "Choice of an installation". Ordinariness does not exist the problem as it is, please push "Next" button.

[About installation directory]
This directory only for High-performance Embedded Workshop. The directory of CC32R (C/C++ Compiler) will be inputted by the installer that run after the operations.

4) Please check the softwares you want, and push "Install" button.
In the default, the main compiler "M32R Toolchains V.X.XX Release xx" and "Autoupdate" are checked.

[What is the AutoUpdate Utility?]
The AutoUpdater will start and station into PC automatically.
The AutoUpdater is an utility that watch the Renesas HomePage periodically and detects the renewal of the installed development tools.

5) "Do you wish to install a new High-performance embedded workshop ?" will be displayed. Please push "Yes" button.

The installers were selected would run in order. → jump to the step 3.

B. In the case of High-performance Embedded Workshop already installed:

1) Push "Installation" button.
2) Select "Update the active High-performance embedded workshop (Recommend)", and push "Next" button.
3) The installed list in present were displayed. Push "Next" button.
4) Please check the software you want, and push "Install" button. In the default, the main compiler "M32R Toolchains V.X.XX Release xx" and "Autoupdate Utility" are checked.

[What is the AutoUpdate Utility?]
The AutoUpdater will start and station into PC automatically.
The AutoUpdater is an utility that watch the Renesas HomePage periodically and detects the renewal of the installed development tools.

5) "Do you wish to update the active High-performance embedded workshop ?" will be

displayed. Please push "Yes" button.

The installers were selected would run in order. → jump to the step 3.

3. Please follow the each installer's messages.

[About target directory of installation]

The install directory of the CC32R (C/C++ Compiler) could be inputted in the installer that was launched in the step. Its default is "C:\Renesas\CC32R\v501r00".

[Constitution of start menu]

After installation, the folders and shortcuts that showed them below will be registered to the [start]->[Programs]->[Renesas].

High-performance Embedded Workshop

M3T-CC32R V.x.xx Release xx (Numbers or versions are displayed to "x".)

Renesas Tools HomePage

4. It is return to the install manager.

Push [Exit] button to end the install manager.

3.2 How to uninstall

The program that became unnecessary can be eliminated from PC by the following method.

1. Confirm whether or not the AutoUpdate icon is on the Windows task tray.
   If its icon is exist, choose by the right click, and select "(E)nd" to finish the AutoUpdate .
2. Open the "Add and remove programs" (or "Add and remove applications" [**]) from the Windows "Control panel".
3. Select "High-performance embedded workshop" in the "Add and remove programs" list and click the "Modify and remove". (This button may be not the "Modify and remove" but the "Remove" or "Add and remove"[**].)

[**] These differ by Windows.

4. Uninstallation will start. Then wait until "Completed" is displayed and push "OK".

3.3 Setting PC environment

If you use on DOS prompt, please set environment variables like Table 1.

Table 1 Example of setting PC environment variables

(The directory name of the installation is supposed with "C:\Renesas\cc32r\v501r00" .)

Environment names	Example of settings
M32RBIN	SET M32RBIN=C:\Renesas\cc32r\v501r00\bin32R
M32RLIB	SET M32RLIB=C:\Renesas\cc32r\v501r00\lib32R
M32RINC	SET M32RINC=C:\Renesas\cc32r\v501r00\inc32R
M32RTMP	SET M32RTMP=C:\Renesas\cc32r\v501r00\TMP
M32RKIN	SET M32RKIN=euc
M32RKOUT	SET M32RKOUT=euc
Command path	PATH %M32RBIN%;%PATH%

[NOTES]

- In this setup example, products are installed by "C:\Renesas\cc32r\v501r00" of the installer. If you wish to install products in a different directory, change the setup contents to the one you want.
- If you use from High-performance Embedded Workshop you do not need to be set these environment variables.

3.4 How to browse the electronic manual

To see these electronic manuals, use a PDF file displaying program, like a "Adobe Acrobat Reader". So install it in your computer as necessary.

- About the Acrobat Reader
  Download "Adobe Acrobat Reader" from following URL of Adobe Systems Incorporated. The latest Acrobat Reader can be downloaded from this home page. Please refer to the following URL about the environment can be use and the latest information for Acrobat Reader.
  `http://www.adobe.com/`

When using CC32R, there are following two methods for displaying an electronic manual:

1. The electronic manuals are registered in the start menu when installing by default. Choose the necessary electronic manual file from menu
   [Start] → [(All) Programs] →[Renesas] → [M32R Family C,C++ Compiler V.x.xx Release x]
   (x expresses version number etc.)

2. Open by double-clicking an electronic manual file
   The electronic manuals are installed in the below `C:\Renesas\cc32r\v501r00\manual` by installation directory of default (`C:\Renesas\cc32r\v501r00`). Double-click these files to open an electronic manual by Acrobat Reader.It is possible to read manuals, that you open by double-clicking these files (extension `.pdf`).

   The following Table 2 lists the electronic manual files.

Table 2 Electronic manual file names

Languages	Manual names	PDF file names
English	M32R Family C/C++ Compiler Package C/C++ Compiler User's Manual	cc32rue.pdf
	M32R Family C/C++ Compiler Package Assembler User's Manual	as32rue.pdf
	MAP Viewer User's Manual	mapue.pdf
Japanese	M32R Family C/C++ Compiler Package C/C++ Compiler User's Manual	cc32ruj.pdf
	M32R Family C/C++ Compiler Package Assembler User's Manual	as32ruj.pdf
	MAP Viewer User's Manual	mapuj.pdf

[NOTE] (About User's Manual of MAP Viewer)

MAP Viewer can be used even if it combines with CC32R. (In this manual, MAP Viewer is used with NC30WA.)

4. Explanation of the added functions

This chapter explains the added function on this version. They are not in the user's manual.

4.1 Designation
4.2 Pragma Operator
4.3 Variable Argument Macro

4.1 Designation

4.1.1 Abstract

The designation can be included in the initializer of structure, union or array. The designation is described before an initializer constant expression in an initializer-list, and indicates that this initializer is for which element (means a member of structure, union, or a cell of array).

The designation is formed by an equal-symbol (=) and a combination of a member name with a dot (.) and '[' constant ']' (array reference).

The following descriptions are some examples of designation:

```
.member =
[3] =
[3][5].member =
.member[4] =
```

Because an initializer was indicated which element of the object clearly, even if it is complicatedly nested structure or array should be initialized more clearly.

4.1.2 Syntax

designation:
designator... =

designator:
. *member-name*
[*constant-expression*]

The designation is composed by one or more designators list and one equal-symbol(=).

The designator is one of .(dot) and member-name or [*constant-expression*].

The .(dot) member name means a member of a structure or union. And [*constant-expression*] means which number element of an array.

4.1.3 Details

If it is an initializer-list with braces { ... }, the elements of it can have a designation.

For example, there is an { 1, 2, 3 } initializer, you can add designators for one of 1, 2 and 3, or all. To add only 2 and 3, you can describe

{ 1, *designation* 2, *designation* 3 }.

Example-1:

For an array:

```
int d[] = { 1, [1] = 2, [2] = 3 };
```

For a structure:

```
struct { int a, b, c; } e = { 1, .b = 2, .c = 3 };
```

If an initializer-list has layer structure, for example, because the { 2, 3 } is an element of the { 1, { 2, 3 }, 4 }, you can describe

{ 1, *designation* { 2, 3 }, 4 }

Yet, the desigation for { 2, 3 } needs to be a structure or array that has two or more elements.

Example-2:

```
struct { int a, b[2], c; } f = { 1, .b = { 2, 3 }, 4 };
```

To select 2 and 3 directory, both
`{ 1, { designation 2, designation 3 }, 4 }`
and
`{ 1, designation 2, designation 3, 4 }`
description are OK.

Example-3:
```
struct { int a, b[2], c; } g = { 1, { [0] = 2, [1] = 3 }, 4 };
```

Example-4:
```
struct { int a, b[2], c; } h = { 1, .b[0] = 2, .b[1] = 3, 4 };
```

4.1.4 Other examples

Example-5 (Combination of array and structure):
```
struct { int a, b[2], c; } m[] = { [1].a = -1, [2].b = {1,2}, [2].c = 3 };
```

Example-6 (Initialization does not rely on the order):
```
struct { int num1, num2, num3; } n = { .num3 = 1, .num1 = -1 };
```

Example-7 (Designation indicates start of initialization):
```
float p[9] = {0.001, 0.01, 0.1, [6] = 10, 100, 1000};
```

Example-8 (Member initialization of an union)
```
struct w { int a, b, c; };
union { struct w s, t, u; } q = { .t = { 1, 2, 3 } };
```

4.1.5 Notes

If a designator is not fit to real member or array element, that causes error.

Example-9 (Designation Failure):
```
struct { int a, b[2], c; } r = { 1, { .b[0] = 2, .b[1] = 3 }, 4 };
```
Because the designator `.b[0] =` is inside braces of the part of r.b, this designator `.b[0] =` means indicates the non-existence element `r.b.b[0]`.
So, the intialization will causes an error in a compilation.

Example-10 (Duplicated initialization):
```
float s[5] = {0.001, 0.01, 0.1, [2] = 10, 100, 1000};
```
For the same s[2] element, there are two initializers that are 0.1 and 10. In this case, the latest initializer 10 must be valid.

4.2 Pragma Operator

4.2.1 Abstract

The pragma operator has same function from the `#pragma` directive, but it was described as same condition as a macro calling.
It is convenient for operating any of the `#pragma` directive functions by using macro.

4.2.2 Syntax

The pragma operator accepts only one string literal as an argument like as follows.

`_Pragma(`*string-literal*`)`

4.2.3 Details

The pragma operator is processed as same as functional-macro, and replaced to same meaning some `#pragma` directives.
For example, `_Pragma("INTERRUPT func")` has same effect of `#pragma INTERRUPT func`.

Example-1:
```
_Pragma("INTERRUPT func")  /* Process as #pragma INTERRUPT func */
void func(void) { }
```

Furthermore, if pragma operator was written on the middle of a line, the replaced `#pragma` directives was appended new-line characters on up and down.

Example-2:
```
_Pragma("SECTION B B1") int v2;
```
→ It was replaced as follows:
```
#pragma SECTION B B1
int v2;
```

4.2.4 Another example

Example-3 (Functional pragma macro):

Definition variable v with t type and a address.
```
#define MAKE_IOPORT(v,t,a) PRAGMA_OPR(ADDRESS v a) t v
#define PRAGMA_OPR(x) _Pragma(#x)
MAKE_IOPORT(p4, unsigned short, 0x123400);
```
→ It was replaced as follows:
```
#pragma ADDRESS p4 0x123400
unsigned short p4;
```

4.2.5 Notes

The argument of a pragma operator can not be omitted. Furthermore, it can not be written two or more strings.

× `_Pragma()`
× `_Pragma("INTERRUPT" "func")`
× `_Pragma("ADDRESS", "a", 0x12340000)`

No appending a semi-colon to the end of the operator.

× `_Pragma("INTERRUPT func");`

4.3 Variable argument macro

4.3.1 Abstract

The macro that has uncertain number of argument can be declarated.

4.3.2 Syntax

`#define macro-name(...)` *replacement-list*

or

`#define macro-name(`*parameters*`, ... )` *replacement-list*

4.3.3 Details

You can use the special form `...` in the macro definition. The other macro or comma must not be described at the right side of the `...`.

The arguments will take in order of the from left to right into the paramters.

So remained arguments will take in to the ... , and next, these argumen texpand to the __VA_ARGS__.

Example-1:
```
#define err_printf(fmt, ...) fprintf(stderr, fmt, __VA_ARGS__)
err_printf("Error: %s in line %d\n", file, line);
```
→ It was replaced as follows:
```
fprintf(stderr, "Error: %s in line %d\n", file, line);
```

If the paramter is only . . . , the all arguments will take in to the . . . .

Example-2:

```
#define mac2(...) int __VA_ARGS__
mac2(v21, v22, v23, v24);
```

→ It was replaced as follows:

```
int v21, v22, v23, v24;
```

4.3.4 Notes

- The parameter other than the right side of a function type macro, can not be the three-dot (. . .) type.
- The __VA_ARGS__ is only in the replacement-list of the macro that has three-dot type parameter.

5. Restrictions on Usage

There are restrictions of the CC32R.

■ On using a control or iteration statement containing a substatement of 128 KB or more in code size. (C/C++ Compiler)

Published to RENESAS TOOL NEWS on August 1, 2007: 070801/tn2

If a control statement (an if statement, for example) or an iteration statement (a while statement, for example) contains a substatement to be executed that consumes a large amount of memory, compiling C source code including those statements may cause the following error to arise, and compilation be unsuccessfully terminated:

```
a132R: "xxx": error: 16-bit displacement overflow in operand 2
```

NOTICE:
"xxx" is the file name given automatically at compilation by the compiler. It is not the name of the C/C++ source file to be compiled.

Conditions:

This problem may occur if the following conditions are all satisfied:

(1) In a control or iteration statement exists a substatement of 128 KB or more in code size.

(2) The substatement in (1) is contained to be executed within any of the following statements:

(a) an if statement; valid only when it executes the substatement in (1) if the result of evaluation of the controlling expression is TRUE

(b) a switch statement; valid only when the total size of the substatements exceeds 128 KB

(c) a while statement

(d) a do statement

(e) a for statement

Examples:

[sample1.c]

```
-------------------------------------------------------------------------
int ans1;
void
func1(int a)
{
   if (a) {                       /* Condition (2)-(a) */
    . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
     /* This part reaches 128 KB */   /* Condition (1) */
    . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   } else {
      ans1 = 1;
   }
}
-------------------------------------------------------------------------
```

[sample2.c]

```
-------------------------------------------------------------------------
int ans2;
void
func2(int a)
{
   switch (a) {                  /* Condition (2)-(b) */
     case 0:
    . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
     /* This part reaches 128 KB */   /* Condition (1) */
```

```
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
       break;
     case 1:
       ans2 = 2;
       break;
   }
}
-------------------------------------------------------------------------
```

[sample3.c]

```
-------------------------------------------------------------------------
void
func3(int a)
{
   while (--a) {                  /* Condition (2)-(c) */
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
     /* This part reaches 128 KB */   /* Condition (1) */
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   }
}
-------------------------------------------------------------------------
```

Solutions:

Avoid this problem in either of the following ways:

(1) In the case of an if statement

Because this problem does not occur if the substatement concerned is placed after else, invert the meaning of the controlling expression and change the places between the substatement concerned and the one after else. When the substatement executed if TRUE is empty after the change, this workaround has no effect.
So be sure to fill it with any statement.

Modification of Example 1 (source file sample1.c)

[sample1.c (After modified)]

```
 ---------------------------------------------------------------
int ans1;
void
func1(int a)
{
  if (!a) {        /* Meaning of controlling expression inverted */
     ans1 = 1;     /* Substatement changed */
  } else {
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   /* This part reaches 128 KB */   /* Substatement changed */
 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
  }
}
 ---------------------------------------------------------------
```

(2) In the case of any control or iteration statement

Convert the whole substatement or a part of the substatement into a newly created function and call it. Be sure not to declare the function to be inline.

Modification of Example 1 (source file sample3.c)

[sample3.c (After modified)]

```
 ---------------------------------------------------------------
void
sub_func3(void)         /* Newly created function */
{
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   /* This part reaches 128 KB */
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
}
void
func3(int a)
```

```
{
   while (--a) {
     sub_func3();    /* Newly created function is called */
   }
}
 ----------------------------------------------------------------
```

■ About describing over 20,000 arguments into the cc32R command line (cc32R compile driver)

The over 20,000 arguments can not be described into the command line of cc32R compile driver.

If the number of described arguments exceed 20,000, cc32R will cause following error.

```
<command line>: error: too many arguments
```

The linking an extremely big application that has nearly 20,000 objects may correspond to such restriction. (It includes a build in the High-performance Embedded Workshop workspace.)

However, the linking by the lnk32R linker has not such limitation.

For avoiding it, the number of the designated object at same time needs to be decreased.

For an example, a certain solution is available:
Divide objects to several groups, make libraries of each object groups (by the lib32R librarian), and describe these libraries into input of the linking.

■ Restriction of making a library that have over 16M bytes. (Librarian)

In CC32R, the size of a library is limited to 16M bytes.

If an operation that is for making a library file more than 16M byte is executed, the librarian lib32R or compiler cc32r will display the following error message.

```
lib32R: error: overflow total module size (>16Mbytes)
```

For solution, divide some library files, and describe all of the divided library files into the linker cc32R.

Furthermore, because of the same reason, the library can't have a object that is over 16M byte. So, in this case, describe this object into the command line of linker directly.

■ About the CPU Selection and the start-up file in the High-performance Embedded Workshop (Integrated Environment)

In the case of making a new workspace of the integrated environment High-performance Embedded Workshop, the "CPU Series" item has no effect to the start-up file. Add the control register setting program needed for the generated start-up file. These settings are written to the user's manuals of each CPU.

Furthermore, the "CPU Series" and "CPU Types" items at making a new workspace will be used to the initial setting of the compiling options.

■ About changing CTOR, VTBL and COMMON sections (C/C++ Compiler)

The CTOR, VTBL and COMMON sections that was made on a C++ compiling

cannot be changed even if one of #pragma SECTION and even -R option was used.

That is because these sections need distinction especially by the standard library and the linker, also if these section names were changed an obstruction on their processing will appear. Therefore, those names can not be changed.

■ Restriction of **`#pragma INTERRUPT`** (**`INTF`**) on the C++ language (C/C++ Compiler)

The `#pragma INTERRUPT` (and the `#pragma INTF`) designates to an interrupt handling

function can be written on a C++ language. But, the functions that were designated shall be of the C language.

Therefore, on the C++ language, the function that is designated with the `#pragma INTERRUPT` shall be declared by `extern "C"` description.

On the case of the very long identifier definition on the C++ language

In the C++ language the result of the 241 or more characters length identifier definition differs from the C language.

Table 3 The compilation results of identifier that has 241 characters or more

Identifiers	In the C language	In the C++ language
global objects	Using only the first 240 characters, and discarding from 241st the characters of the back.	Using only the first 240 characters, and discarding from 241st the characters of the back.
Other identifiers (function, class, structure, union, member, argument)	Using only the first 240 characters, and discarding from 241st the characters of the back.	The compiler recognizes and distinguishes from 241st the characters of the back, but these names are symbolized for the recognizing. Therefore, for example, a function name is not displayed by the original name in the linking error, etc.

On the case of describing outside range value to the **`#line`** directive in the C++ language

In the C++ language the result of describing outside range value to the `#line` directive differs from the C language.

Caution:

Even if the language which is C or C++ in CC32R, the 32,768 or more value cannot be described to the #line directive. Actually, even without defending this regulation, the generated M32R machine codes will work. However, be careful as for that an compiling error may occur, and the source file may not be displayed normally when the generated object is used for debugging.

1 ~ 32,767:

It is processed normally.

32,768 ~ 4,294,967,295:

An internal error may happen as follows.

```
internal error: BAD Location
internal error: unreachable logic
```

4,294,967,296 or more:

It is processed as an error as follows

```
error: invalid line number
```

6. Precautions on using of V.5.01 Release 00

There are precautions on usage about this version.

■ On using a function-like macro that takes the same function-like macro as an argument, the compiler may not expand the same function-like macro. (C Compiler)

Published to RENESAS TOOL NEWS on September 1, 2008: 080901/tn3

If a function-like macro that is spread over two or more lines takes the same function-like macro as an argument, the latter may not be expanded:

Conditions:

This problem occurs if the following conditions are all satisfied:

(1) A function-like macro statement is spread over two or more lines.

(2) The function-like macro in (1) takes the same function-like macro as an argument.

(3) The function-like macro as an argument in (2) is placed in front of a line or just follows comments or a sequence of white-space characters in front of a line.

Examples:

```
[sample1.c]
-----------------------------------------------------------------------
#define SMP_MACRO(a,b) (a + b)
int var = SMP_MACRO(1,                <- Condition (1)
          SMP_MACRO(2,3));           <- Conditions (2) and (3)
-----------------------------------------------------------------------

Result of macro replacement (output of cc32R -E sample1.c):
-----------------------------------------------------------------------
#line 1 "sample1.c"

int var = (1 +     SMP_MACRO(2,3)) ; /* SMP_MACRO(2,3) not expanded */
-----------------------------------------------------------------------
```

Solutions:

Avoid the problem in either of the following ways:

(1) Concatenate the line containing the function-like macro as an argument in (3) to the preceding line by placing a \ character at the end of the preceding line.

```
Modification of sample1.c:
  -----------------------------------------------------------------------
  #define SMP_MACRO(a,b) (a + b)
  int var = SMP_MACRO(1,        \  <- \ placed at the end of this line
            SMP_MACRO(2,3));
  -----------------------------------------------------------------------
```

(2) Define another function-like macro with the same format and a different name; then use it as an argument.

```
Modification of sample1.c:
  -----------------------------------------------------------------------
  #define SMP_MACRO(a,b) (a + b)
  #define SMP_SAME_MACRO(a,b) (a + b)    <- Same format as SMP_MACRO
  int var = SMP_MACRO(1,
            SMP_SAME_MACRO(2,3));      <- SMP_MACRO replaced
  -----------------------------------------------------------------------
```

■ On making a function-like macro statement that extends over more than one line and uses comments beginning with // or enclosed with /* and */ the compiler may display an error. (C Compiler)

Published to RENESAS TOOL NEWS on July 1, 2008: 080701/tn1

If a function-like macro statement is made which extends over more than one line and uses comment statements beginning with / / or enclosed with /* and */, the following error message may be dispatched during compilation:

```
Error message:
-------------------------------------------------------------------------------
    error: unterminated comment
-------------------------------------------------------------------------------
```

Conditions:

This problem may occur if the following conditions are all satisfied:

(1) A function-like macro is stated extending over more than one line.

(2) In the ending line of the macro in (1), a comment statement beginning with / / or enclosed with /* and */ is placed.

(3) In the line immediately before the line in (2), another comment statement beginning with / / is placed.

(4) The size of the C source file concerned is greater than 512 bytes. Note that the size includes that of the include files if they are.

Example:

```
-------------------------------------------------------------------
#define SAMPLE_MACRO(a,b,c)  (a + b + c)
   /* This program part omitted */
int func(void)
{
   return SAMPLE_MACRO( 1,
                    2,   // No.2    <-- Conditions (1) and (3)
                    3);  // No.3    <-- Conditions (1) and (2)

}
-------------------------------------------------------------------
```

In the above example, the problem may not arise depending on the descriptions of the omitted program part.

Solution:

Delete the comment statement in Condition (3), which begins with / /, or change it to the one enclosed with /* and */.

Modification of sample1.c:

```
  -------------------------------------------------------------------
  return SAMPLE_MACRO( 1,
                   2,   /* No.2 */  <-- Enclosed with /* and */
                   3);  // No.3
  -------------------------------------------------------------------
```

■ On using ten thousand or more initializing expressions, the compiler may not compile normally. (C Compiler)

Published to RENESAS TOOL NEWS on August 1, 2007: 070801/tn1

If ten thousand or more initializing expressions exist in a list of them enclosed with curry braces in a C / C++ source code, M3T-CC32R may malfunction and compilation be terminated with the following Windows' error message dispatched:

```
Windows' error message:
-------------------------------------------------------------------------------
postpar.exe has encountered a problem and needs to close. We are sorry for the inconvenience.
-------------------------------------------------------------------------------
```

Conditions:

In the C++ and the C language, the conditions under which this problem occurs are different from each other.

1. In the C++ Language

This problem may occur if the following conditions are both satisfied:

(1) In the declaration of an array of type short or char (including unsigned or signed attached), there exists a list of initializing expressions containing about ten thousand or more of them.

(2) In lines before and after the list in (1) above, other declarations or functions are placed.

2. In the C Language

This problem may occur if the following conditions are both satisfied:

(1) In the declaration of an array of any type exists a list of initializing expressions containing ten thousand or more of them that are cast in their types.

(2) Either of the following conditions is met:

(a) In the declaration of an array in (1), the number of elements of the array is neglected.

(b) In a line after the list in (1) above, another declaration or a function is placed.

Examples:

1. C++ Language

[sample1.cpp]

```
-----------------------------------------------------------------------
unsigned short table1[] = {       /* Condition 2.1 (1) */
    0,
    112,
    93,
. . . . . .
    /* Thirty thousand initializing expressions */
. . . . . .
};
int
check_table1(int index)          /* Condition 2.1 (2) */
{
    int ans;
    ans = table1[index];
    return ans;
}
-----------------------------------------------------------------------
```

2. C Language

[sample2.c]

```
-----------------------------------------------------------------------
int table2[] = {             /* Conditions 2.2 (1) and (2)-(a) */
    (unsigned) 0,
    (unsigned) 100392,
    (unsigned) 33293,
. . . . . .
    /* Fifty thousand initializing expressions */
. . . . . .
};
-----------------------------------------------------------------------
```

[sample3.c]

```
-----------------------------------------------------------------------
int table3[50003] = {           /* Condition 2.2 (1) */
    (unsigned) 0,
    (unsigned) 100392,
    (unsigned) 33293,
. . . . . .
    /* Fifty thousand initializing expressions */
. . . . . .
};
int
check_table3(int index)          /* Condition 2.2 (2)-(b) */
{
    int ans;
```

```
    ans = table3[index];
    return ans;
}
-----------------------------------------------------------------
```

Command line input

```
-----------------------------------------------------------------
% cc32R -S sample1.cpp
% cc32R -S sample2.c
% cc32R -S sample3.c
 NOTE: “%” denotes a prompt.
-----------------------------------------------------------------
```

Solutions:

Avoid this problem in either of the following ways:

(1) In C++, move the declaration of an array containing the list of initializing expressions to another C++ source file.

Modification of Example 1 (source file sample1.cpp):

Move the declaration of an array containing the list of initializing expressions to a newly created C++ source file, sample1init.cpp, for example; then in the original place, declare the array concerned to be extern with no initializing expressions.

[sample1init.cpp (Newly created)]

```
-----------------------------------------------------------------
unsigned short table1[] = {
  0,
  112,
  93,
. . . . . .
  /* Thirty thousand initializing expressions */
. . . . . .
};
-----------------------------------------------------------------
```

[sample1.cpp (After modified)]

```
-----------------------------------------------------------------
extern unsigned short table1[];
   /* Declared to be extern with no initializing expressions */
int
check_table1(int index)
{
    int ans;
    ans = table1[index];
    return ans;
}
-----------------------------------------------------------------
```

(2) In C, do not cast the types of the initializing expressions.

Modification of Example 2 (source file sample2.c):

Remove all the cast expressions before the initializing expressions.

[sample2.c (After modified)]

```
-----------------------------------------------------------------
int table2[] = {
      0,                    /* Remove cast expressions */
    100392,                  /* As above */
    33293,                   /* As above */
. . . . . .
  /* Fifty thousand initializing expressions */
. . . . . .
};
-----------------------------------------------------------------
```

(3) In C, write the number of elements of the array in its declaration, and move the declaration to the last of the source code.
This is the alternative to Workaround (2) above when it cannot be used.
Note that if Conditions (a) and (b) in 2.2 (2) are met, both the workarounds explained below for sample2.c and sample3.c must be taken.

Modification of Example 2 (source file sample2.c):
Do not neglect the number of elements of array table2 but declare it.

[sample2.c (After modified)]

```
int table2[50003] = {       /* Declare the number of elements */
  (unsigned) 0,
  (unsigned) 100392,
  (unsigned) 33293,
      :
  /* Fifty thousand initializing expressions */
      :
};
```

Modification of Example 3 (source file sample3.c):
Move the declaration of array table3 with a list of initializing expressions to the last of the function that follows, and re-declare the array with no initializing expressions in the original place.

[sample3.c (After modified)]

```
int table3[50003];              /* Declare array only */
int
check_table3(int index)
{
   int ans;
   ans = table3[index];
   return ans;
}
int table3[50003] = {      /* Move original declaration here */
   (unsigned) 0,
   (unsigned) 100392,
   (unsigned) 33293,
      :
  /* Fifty thousand initializing expressions */
      :
};
```

7. Software table

Table 4 shows the look of the directory and file that are made after installation.

Table 4 Table of the directory and file after installation

Directory names	File names	Notes
bin32R	cc32R.exe	Compile driver (V.3.00.03.001)
	as32R.exe	Assemble driver (V.2.05.00.000)
	lnk32R.exe	Linker (V.1.15.01.000)
	lib32R.exe	Librarian (V.1.06.00.000)
	lmc32R.exe	Load module converter (V.1.14.00.000)
	map32R.exe	Map generator (V.1.23.02.000)
	libg32R.exe	Library generator (V.1.00.00.000)
	strip32R.exe	Debug information discarding utility (V.1.02.00.000)
lib32R	cpre.exe	Preprocessor (V.2.09.01.000)
	cpfrt32R.exe	C++ Parser (V.1.03.00.000)
	cfrt.exe	Parser (V.2.29.01.000)
	postpar.exe	Post parser (V.1.04.02.000)
	opt.exe	Optimizer (V.1.29.01.000)
	cg32R.exe	Code generator (V.4.05.01.000)
	genstk.exe	Stack file generator (V.1.00.00.000)
	a032R.exe	Macro processor (V.1.02.00.000)
	a132R.exe	Assembler (V.4.06.01.000)
	alis32R.exe	List processor (V.1.02.00.000)
	parafilt.exe	Parallel processor (V.1.01.00.000)
	plink32R.exe	Pre-linker (V.1.00.00.000)
	conv32R.exe	SYSROF to ELF converter (V.1.02.01.001)
	cmerge.exe	C source merge processor (V.1.03.00.000)
	m32RcR.lib	C Library (Small model)
	m32RcRM.lib	C Library (Medium model)
	m32RcRL.lib	C Library (Large model)
	m32RcR.stk	Stack utilize display file for C library (for m32RcR.lib)
	m32RcRM.stk	Stack utilize display file for C library (for m32RcRM.lib)
	m32RcRL.stk	Stack utilize display file for C library (for m32RcRL.lib)
inc32R	assert.h, ctype.h, errno.h, float.h, limits.h, locale.h, long64.h, math.h, mathf.h, setjmp.h, signal.h, stdarg.h, stddef.h, stdio.h, stdlib.h, string.h, time.h, cstddef, cstdio, cstdlib, exception, new, new_ecpp.h, new_std.h, stdexcept, typeinfo	C Library headers
inc32R\sys	assert.h, ctype.h, errno.h, float.h, limits.h, locale.h, math.h, mathf.h, setjmp.h, signal.h, stdarg.h, stddef.h, stdio.h, stdlib.h, string.h, time.h	System definition headers (*1)
inc32R\com	ANSI_errno.h, def.h, SBPP	
UnSpt32R (*2)	abslist.exe	Absolute listing utility (V.1.02.00.000)
	stk32R.exe	Stack size calculation utility (V.1.02.01.000)
	license.txt	Development support utility guide (English)
	license.sj	Development support utility guide (Japanese)
	abslist.txt	Utility manual of abslist (English)
	stk.txt	Utility manual of stk32R (English)
	abslist.sj	Utility manual of abslist (Japanese)
	stk.sj	Utility manual of stk32R (Japanese)
lib32R	lbg_cc32r.dep	Files for C library generating
lib32R\src	~.cpp ~.c ~.ms	
lib32R\pack	~.pack	

support	mtoolspt.htm	Link-page for "Renesas Technology Software and Tools"
support\cc32r	support.txt regist.txt	Customer and Product information record files
smp32R	start.ms	Startup, Low-level functions example
bin	mapviewer.exe map_inspect.dll mapviewer.hlp mapviewer.cnt	Map Viewer (V.3.00.00) DLL file for Map Viewer Help file for Map Viewer Help setting file for Map Viewer
manual	CC32Rue.pdf AS32Rue.pdf CC32Ruj.pdf AS32Ruj.pdf mapuj.pdf	User's manual < C Compiler > [English] User's manual < C Compiler > [English] User's manual < C Compiler > [Japanese] User's manual < C Compiler > [Japanese] Map Viewer Manual [English or Japanese (will be selected by the installer)]
	~.msf	Manual attribute files
hew	~.det ~.dll ~.tbp	High-performance Embedded Workshop setting files
tmp	<< Empty >>	Temporary directory for CC32R

NOTES

*1) The System definition headers are unable to be deleted and modified, because CC32R refers to them in compiling.
If these files are deleted or changed CC32R does not run normally.

*2) The program in the UnSpt32R directory differ from the constitution things of other CC32R regarding the handling of the license and support. Please confirm the document file "license.txt" in this directory.